PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F

A2

(11) International Publication Number: **WO 99/57620**

(43) International Publication Date: 11 November 1999 (11.11.99)

(21) International Application Number: PCT/FI99/00374

(22) International Filing Date: 4 May 1999 (04.05.99)

(30) Priority Data:
980985 4 May 1998 (04.05.98) FI

(71) Applicant *(for all designated States except US)*: SONERA OY [FI/FI]; Teollisuuskatu 15, FIN-00510 Helsinki (FI).

(72) Inventor; and

(75) Inventor/Applicant *(for US only)*: TÖHÖNEN, Harri [FI/FI]; Porintie 3 A 4, FIN-00350 Helsinki (FI).

(74) Agent: PAPULA REIN LAHTELA OY; Fredrikinkatu 61 A, P.O. Box 981, FIN-00100 Helsinki (FI).

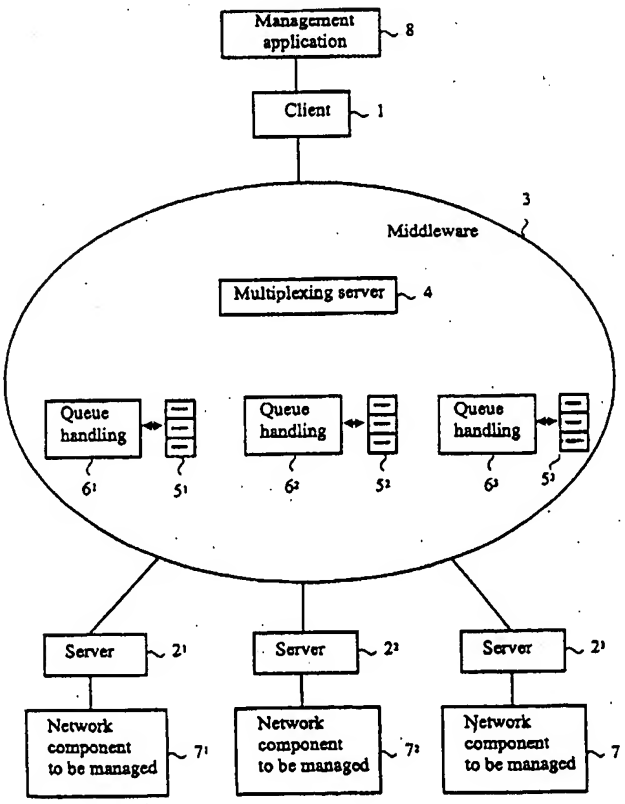
(81) Designated States: AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, US, UZ, VN, YU, ZA, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published
Without international search report and to be republished upon receipt of that report.

(54) Title: DISTRIBUTION OF A SERVICE REQUEST

(57) Abstract

The present invention relates to a system and a procedure for distribution of a service request in a computer system based on a client–server architecture and comprising at least one client (1), which sends service requests to servers (2^1, 2^2, ..., 2^n); one or more servers (2^1, 2^2, ..., 2^n), which provide callable services for the client (1); and middleware means (3). According to the invention, the system comprises a multiplexing server (4), by means of which the client's (1) service request is copied and distributed to the servers (2^1, 2^2, ..., 2^n) for parallel execution; a queue system (5^1, 5^2, ..., 5^n) for storing service requests not yet executed; and queue handling means (6^1, 6^2, ..., 6^n), by means of which the service requests placed in the queue system (5^1, 5^2, ..., 5^n) are passed on for re-execution. The invention allows transparent distribution of the service request, guarantees improved reliability of operation in failure situations and consistency of resources.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece	ML	Mali	TR	Turkey
BG	Bulgaria	HU	Hungary	MN	Mongolia	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MR	Mauritania	UA	Ukraine
BR	Brazil	IL	Israel	MW	Malawi	UG	Uganda
BY	Belarus	IS	Iceland	MX	Mexico	US	United States of America
CA	Canada	IT	Italy	NE	Niger	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NL	Netherlands	VN	Viet Nam
CG	Congo	KE	Kenya	NO	Norway	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NZ	New Zealand	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	PL	Poland		
CM	Cameroon	KR	Republic of Korea	PT	Portugal		
CN	China	KZ	Kazakstan	RO	Romania		
CU	Cuba	LC	Saint Lucia	RU	Russian Federation		
CZ	Czech Republic	LI	Liechtenstein	SD	Sudan		
DE	Germany	LK	Sri Lanka	SE	Sweden		
DK	Denmark	LR	Liberia	SG	Singapore		
EE	Estonia						

DISTRIBUTION OF A SERVICE REQUEST

BACKGROUND OF THE INVENTION

The present invention relates to a system as defined in the preamble of claim 1 for distribution of a service request in a computer system based on a client-server architecture and to a procedure as defined in the preamble of claim 8 for distribution of a service request in a computer system based on a client-server architecture.

In a three-level client-server architecture, the server/servers produces/produce services requested by the client/clients. An agent acting between these is middleware, which is used to take care of e.g. the routing of service requests to the appropriate server, equalisation of load and management of transactions.

A typical area of application is telecommunication network management. In this case, the client application is e.g. a customer management system and the server application represents e.g. an SCP intelligent network component (Service Control Point, SCP). When the operation of the SCP intelligent network component, i.e. the node to be managed, is to be rendered as reliable as possible, the node is physically doubled, so that both nodes contain the same service data. The duplication must not be visible to the client process, but changes made by the client in the service data must be copied to both nodes. In other words, when the client sends a management data update request, the update request must be transmitted to all the servers and the management data must be updated in all the nodes to be managed, but the client must perceive the entire transaction as if it were only dealing with a single server/node.

In prior-art systems, a service request can be distributed in a client application separately to

each identical resource. Thus, the distribution architecture is visible to the client application. Further, if an update operation in one of the nodes to be managed is unsuccessful e.g. due to a telecommunication problem, the result is that the nodes are in a non-consistent state because the update cannot be cancelled automatically from the middleware level.

In middleware environments, a two-stage commitment procedure is previously known. In such a procedure, the changes caused by a service request are first saved temporarily, whereupon a transaction monitor sends a preliminary commitment command (pre-commit) to each server. If the transaction monitor receives a confirmation from each server, then it sends the servers an actual commitment command (commit), and the changes caused by the service request are updated simultaneously. If no confirmation is received from one of the servers, then no commitment command is sent and the changes are not updated, thus preserving the consistency of resources.

Two-stage commitment is in itself a workable procedure. The problem is that not all of the nodes to be managed are capable of it. In other words, a service request can only be cancelled if the cancellation decision is made from the client application. As a result, however, the distribution is visible to the client.

SUMMARY OF THE INVENTION

The object of the present invention is to disclose a new type of procedure and system for eliminating the above-mentioned drawbacks.

A specific object of the present invention is to disclose a solution for achieving transparent distribution of a service request and consistency of resources.

As for the features characteristic of the present invention, reference is made to the claims.

The system of the invention for distribution of a service request in a computer system based on a client-server architecture comprises at least one client application/client process which sends service requests to servers; one or more server applications/server processes providing services which can be called by the client; and middleware means; furthermore, the system comprises a multiplexing server, a queue system and queue handling means, which are implemented e.g. as software components. For each server, preferably a separate queue system and queue handling means are provided. This ensures maximal reliability in failure situations. The multiplexing server provides the same service interface as the actual servers, and the names of the services of the actual servers are changed, causing the clients' service calls to be directed to the multiplexing server.

According to the invention, by means of the multiplexing server, the client's service request is copied and distributed to the servers for execution. By means of the multiplexing server, responses are obtained from the servers and a single response for the calling client is composed from them. If any one of the servers returns a successful response, then a successful response is returned to the client. Thus, the duplication is hidden from the client. For those servers which return an unsuccessful response due e.g. to a telecommunication failure preventing communication with the resource to be managed, the service request is placed in the queue system for the server in question. The multiplexing server does not take care of service requests placed in a queue system but is ready to serve subsequent service requests of the client/clients.

Further, according to the invention, service requests not fulfilled are stored in server-specific queue systems, which comprise a separate queue for each service provided by the server. The queues are based on prioritisation of requests, i.e. a service request entered in a queue with the highest priority is the first one to be served and removed from the queue. The queue system is implemented e.g. in the form of a file.

Further, according to the invention, the queue handling means are used to pass on the service requests placed in the queue system to the servers at predetermined time intervals for re-execution. For example, a list of the queues to be monitored is maintained. A service request is taken from the queue, and the server service corresponding to the queue is called.

As compared with prior art, the present invention has the advantage that it allows transparent distribution of a service request, in other words, a client's service request can be copied to a plurality of servers without the copying being visible to the client. Furthermore, according to the invention, instead of cancellation of a transaction, a queue mechanism is used whereby attempts to execute the transaction are repeated until the fault has been corrected and the transaction is successfully carried out or until a time limit is exceeded. This makes it possible to achieve a greater reliability in failure situations than before. Moreover, the invention requires only minimal changes in the existing middleware architecture as regards the client and server processes, so it can be easily implemented and taken into use.

In an embodiment of the invention, the multiplexing server, the queue system and the queue handling means are logically disposed in conjunction with the existing middleware means between the client and

the servers. Each server communicates with a given telecommunication network component, which network components are managed by a management application communicating with the client. The servers provide identical services to the client, with the difference that different servers communicate with different manifestations of the network components to be managed.

In an embodiment of the invention, a table of the servers to which the service request is to be duplicated is maintained in the multiplexing server. The services of the servers in the table are preferably called asynchronically by the multiplexing server, in other words, having transmitted a service request to one server, the multiplexing server starts transmitting the service request to the next server without waiting for a response from the first server. This allows parallel execution without delays.

In an embodiment of the invention, before calling a service, a check is carried out on each server by the multiplexing server to determine whether there are already any service requests in the queues corresponding to the service in question. If the queue concerned already contains service requests, then the new service request is placed directly in the appropriate queue and assigned the lowest priority at the moment.

In an embodiment of the invention, a cycle counter and/or a time stamp are/is attached to service requests transferred into a queue system to allow control of the length of time each service request is retained in the queue system.

In an embodiment of the invention, to retrieve a service request from a queue by using queue handling means, the server service corresponding to the queue is called synchronically, in other words, a response to the call is awaited. If a message indicat-

ing successful execution of the service request is received from the server, then the service request has been successfully executed. If the server is still unable to execute the service request, then the service request is again placed in the queue and the cycle counter for the service is incremented. However, if the time stamp and/or cycle counter of the service request exceed certain predetermined values, then the service request is deleted from the queue and the situation is logged.

In an embodiment of the invention, the middleware means consist of Tuxedo middleware.

BRIEF DESCRIPTION OF THE DRAWINGS

In the following, the invention will be described by the aid of a few examples of its embodiments by referring to the attached drawing, wherein

Fig. 1 presents an embodiment of the system of the invention, and

Fig. 2 presents a functional diagram of a system according to the invention.

DESCRIPTION OF THE PREFERRED EMBODIMENT

Fig. 1 presents an embodiment of the system of the invention for implementing telecommunication network management in which a management application 8 is used to manage components $7^1, 7^2, 7^3$ of the telecommunication network. In this example, the network components are SCP nodes in an intelligent network, nodes 7^2 and 7^3 being backup copies of node 7^1. Nodes 7^2 and 7^3 thus contain the same intelligent network service data as node 7^1. A notable feature is that the SCP nodes $7^1, 7^2, 7^3$ are incapable of two-stage commitment to data updates. A management application 8 communicates with a client process 1, and the SCP nodes $7^1, 7^2, 7^3$ communicate with server processes $2^1, 2^2, 2^3$. Functioning be-

tween the client process 1 and the server processes $2^1, 2^2, 2^3$ is Tuxedo middleware application management software 3, which takes care of e.g. routing of the service requests to the appropriate server, load equalisation and transaction management. Implemented as logical software components disposed between the client process 1 and the server processes $2^1, 2^2, 2^3$ are a multiplexing server 4, queue handling means $6^1, 6^2, 6^3$ and queue systems $5^1, 5^2, 5^3$. For each server, there is one queue handling means and one queue system.

Fig. 2 presents a functional diagram of a system according to the invention in a Tuxedo environment. By means of a client application (not shown), a call for service y is issued, which service is actually implemented as service y' in server A 2^1 and server B 2^2. First, configuration information is read from a run-time management information base (MIB) in Tuxedo 3. The configuration information comprises e.g. server groups, routing data for the groups, queue parameters and queue space data. When the service request y is received by the multiplexing server 4, a check is carried out to establish whether the queue systems 5^1 and 5^2 (not shown) for servers 2^1 and 2^2 already contain any messages. In this way, the order of the service requests is maintained. The name of the service is changed to y'. A routing field is added to the message.

Next, a call for service y' is sent asynchronically to each application server 2^1 and 2^2. Before the call, the routing field is assigned a value which is obtained from a table maintained by the multiplexing server 4, said table containing routing field values for the servers 2^1 and 2^2. The values were obtained from the Tuxedo 3 configuration information when the multiplexing server 4 was started up. Thus, the service requests are routed to different application servers. Next, responses are obtained from the

application servers 2^1 and 2^2. It is to be noted that Tuxedo automatically collects the responses to asynchronous calls. The title part of the message is checked for errors, and the results are added to a results table which contains table locations for each application server 2^1 and 2^2. Based on the results table, a response message for the client application is composed.

The results table is analysed. If errors are detected e.g. in the response of application server 2^1, then a time stamp and a cycle counter are attached to the message concerned. Further, the lowest priority in the queue system 5^1 corresponding to the application server 2^1 is found out and the message is assigned the next priority level below it. After this, the message is placed in the queue system 5^1 for the application server 2^1.

The queues are polled using the queue handling means 6^1 and 6^2 (not shown). For each service, there are specified queues in the queue systems 5^1 and 5^2. For each application server 2^1, 2^2 there is one queue system 5^1, 5^2 and one set of queue handling means 6^1, 6^2. Using queue handling means 6^1, a call for service y' is issued synchronously. Before the call, the time stamp and cycle counter are checked. If the time stamp is too old and the cycle counter has a sufficient value, then the message is deleted from the queue system 5^1, and the situation is logged. If an erroneous response is still obtained from the application server 2^1, then the message is placed back into the queue and the cycle counter value for the message is incremented.

The invention is not restricted to the examples of its embodiments described above, but instead many variations are possible within the scope of the inventive idea defined in the claims.

CLAIMS

1. System for distribution of a service request in a data system based on a client-server architecture and comprising at least one client (1), which sends service requests to servers $(2^1,2^2,...,2^n)$; one or more servers $(2^1,2^2,...,2^n)$, which provide callable services for the client (1); and middleware means (3), characterised in that the system comprises

- a multiplexing server (4), by means of which the client's (1) service request is copied and distributed to the servers $(2^1,2^2,...,2^n)$ for parallel execution, the client (1) is notified of successful/unsuccessful execution of the service request, and if one of the servers $(2^1,2^2,...,2^n)$ is unable to execute the service request, then the service request is transferred into the queue system $(5^1,5^2,...,5^n)$ for the server concerned;

- a queue system $(5^1,5^2,...,5^n)$ for each server $(2^1,2^2,...,2^n)$ for storing service requests not yet executed; and

- queue handling means $(6^1,6^2,...,6^n)$ for each server $(2^1,2^2,...,2^n)$, said means being used to pass on service requests placed in the queue system $(5^1,5^2,...,5^n)$ for re-execution at certain time intervals.

2. System as defined in claim 1, characterised in that

- the multiplexing server (4), the queue system (51,52,...,5n) and the queue handling means (61,62,...,6n) are disposed in conjunction with the middleware means (3) between the client (1) and the servers (21,22,...,2n);

- each server (21,22,...,2n) is connected to a telecommunication network component (71,72,...,7n), which network components (71,72,...,7n) are managed using a management application (8) communicating with the client (1); and

- the servers (21,22,...,2n) provide identical services for the client (1).

3. System as defined in claim 1 or 2, characterised in that

- a table of the servers $(2^1,2^2,...,2^n)$ to which the service request is to be copied is maintained in the multiplexing server (4); and

- the services of the servers $(2^1,2^2,...,2^n)$ are called asynchronically by the multiplexing server (4).

4. System as defined in any one of the preceding claims 1 - 3, characterised in that

- before issuing a call for a service, a check is carried out on each server by the multiplexing server (4) to determine whether there are any service requests in the queue systems $(5^1,5^2,...,5^n)$; and

- if there are already service requests in the queue system $(5^1,5^2,...,5^n)$, then the service request is placed in the queue system $(5^1,5^2,...,5^n)$ by the multiplexing server (4) and assigned the lowest priority at the moment.

5. System as defined in any one of the preceding claims 1 - 4, characterised in that a cycle counter and/or a time stamp are/is attached to the service requests transferred into a queue system $(5^1,5^2,...,5^n)$ by the multiplexing server (4).

6. System as defined in any one of the preceding claims 1 - 5, characterised in that

- using the queue handling means $(6^1,6^2,...,6^n)$, the services of the server $(2^1,2^2,...,2^n)$ are called synchronically;

- if the server $(2^1,2^2,...,2^n)$ is unable to carry out the service, then the service request is returned to the queue system $(5^1,5^2,...,5^n)$ by the queue handling means $(6^1,6^2,...,6^n)$ and the cycle counter for the service request is incremented; and

- if the time stamp and/or cycle counter of the service request exceed certain predetermined values,

then the service request is deleted by the queue handling means ($6^1, 6^2, \ldots, 6^n$).

7. System as defined in any one of claims 1 - 6, characterised in that the middleware means (3) consist of Tuxedo middleware.

8. Procedure for distribution of a service request in a data system based on a client-server architecture and comprising at least one client (1); one or more servers ($2^1, 2^2, \ldots, 2^n$); and middleware means (3), in which procedure service requests are sent from the client (1) to the servers ($2^1, 2^2, \ldots, 2^n$), and services called are provided for the clients (1) by the servers ($2^1, 2^2, \ldots, 2^n$), characterised in that

- by means of a multiplexing server (4), the client's (1) service request is copied and distributed to the servers ($2^1, 2^2, \ldots, 2^n$) for parallel execution, the client (1) is informed about successful/unsuccessful execution of the service request, and if one of the servers ($2^1, 2^2, \ldots, 2^n$) is unable to carry out the service request, then the service request is transferred into a queue system ($5^1, 5^2, \ldots, 5^n$) for the server concerned;

- service requests not yet executed are stored in server-specific queue systems ($5^1, 5^2, \ldots, 5^n$); and

- using server-specific queue handling means ($6^1, 6^2, \ldots, 6^n$), service requests placed in the queue system ($5^1, 5^2, \ldots, 5^n$) are passed on for re-execution at certain time intervals.

9. Procedure as defined in claim 9, characterised in that

- the multiplexing server (4), the queue system ($5^1, 5^2, \ldots, 5^n$) and the queue handling means (6) are disposed in conjunction with the middleware means (3) between the client (1) and the servers ($2^1, 2^2, \ldots, 2^n$);

- each server ($2^1, 2^2, \ldots, 2^n$) is connected to a telecommunication network component ($7^1, 7^2, \ldots, 7^n$), which network components ($7^1, 7^2, \ldots, 7^n$) are managed by

a management application (8) communicating with the client (1); and

- identical services are provided for the client (1) by the servers ($2^1, 2^2, \ldots, 2^n$).

10. Procedure as defined in claim 8 or 9, characterised in that

- a table of the servers ($2^1, 2^2, \ldots, 2^n$) to which the service request is to be copied is maintained in the multiplexing server (4); and

- the services of the servers ($2^1, 2^2, \ldots, 2^n$) are called asynchronically by the multiplexing server (4).

11. Procedure as defined in any one of the preceding claims 8 - 10, characterised in that

- before issuing a call for a service, a check is carried out on each server by the multiplexing server (4) to determine whether there are any service requests in the queue systems ($5^1, 5^2, \ldots, 5^n$); and

- if there are already service requests in the queue system ($5^1, 5^2, \ldots, 5^n$), then the service request is placed in the queue system ($5^1, 5^2, \ldots, 5^n$) by the multiplexing server (4) and assigned the lowest priority at the moment.

12. Procedure as defined in any one of the preceding claims 8 - 11, characterised in that a cycle counter and/or a time stamp are/is attached to service requests transferred into a queue system ($5^1, 5^2, \ldots, 5^n$) by the multiplexing server (4).

13. Procedure as defined in any one of the preceding claims 8 - 12, characterised in that

- using the queue handling means ($6^1, 6^2, \ldots, 6^n$), the services of the server ($2^1, 2^2, \ldots, 2^n$) are called synchronically;

- if the server ($2^1, 2^2, \ldots, 2^n$) is unable to carry out the service, then the service request is returned to the queue system ($5^1, 5^2, \ldots, 5^n$) by the queue handling means ($6^1, 6^2, \ldots, 6^n$) and the cycle counter for the service request is incremented; and

- if the time stamp and/or cycle counter of the service request exceed certain predetermined values, then the service request is deleted by the queue handling means ($6^1, 6^2, \ldots, 6^n$).

14. Procedure as defined in any one of the preceding claims 8 - 13, characterised in that the middleware means (3) consist of Tuxedo middleware.

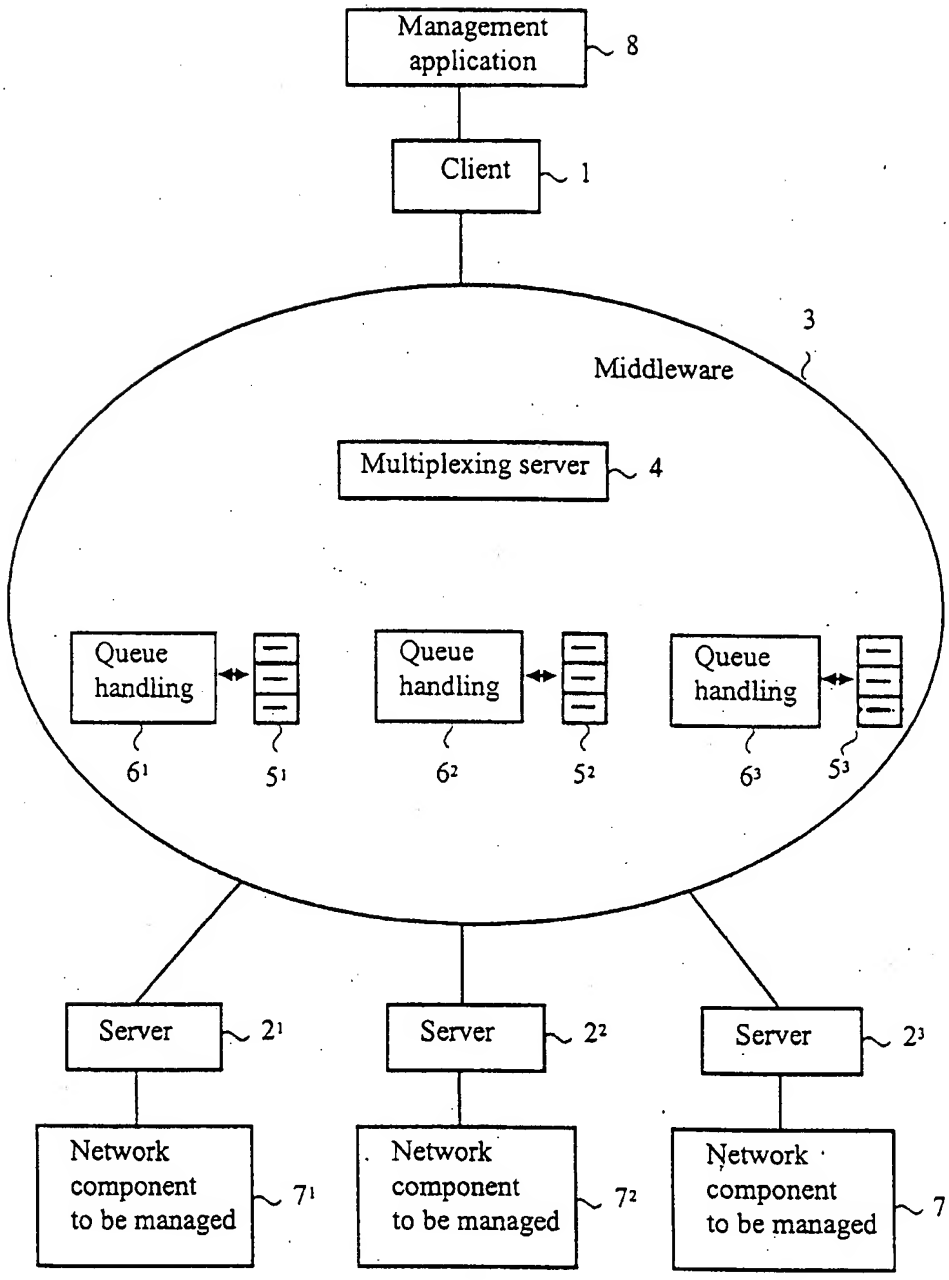

Management application
8
Client
1
3
Middleware
Multiplexing server
4
Queue handling
6¹
5¹
Queue handling
6²
5²
Queue handling
6³
5³
Server
2¹
Server
2²
Server
2³
Network component to be managed
7¹
Network component to be managed
7²
Network component to be managed
7

Fig. 1

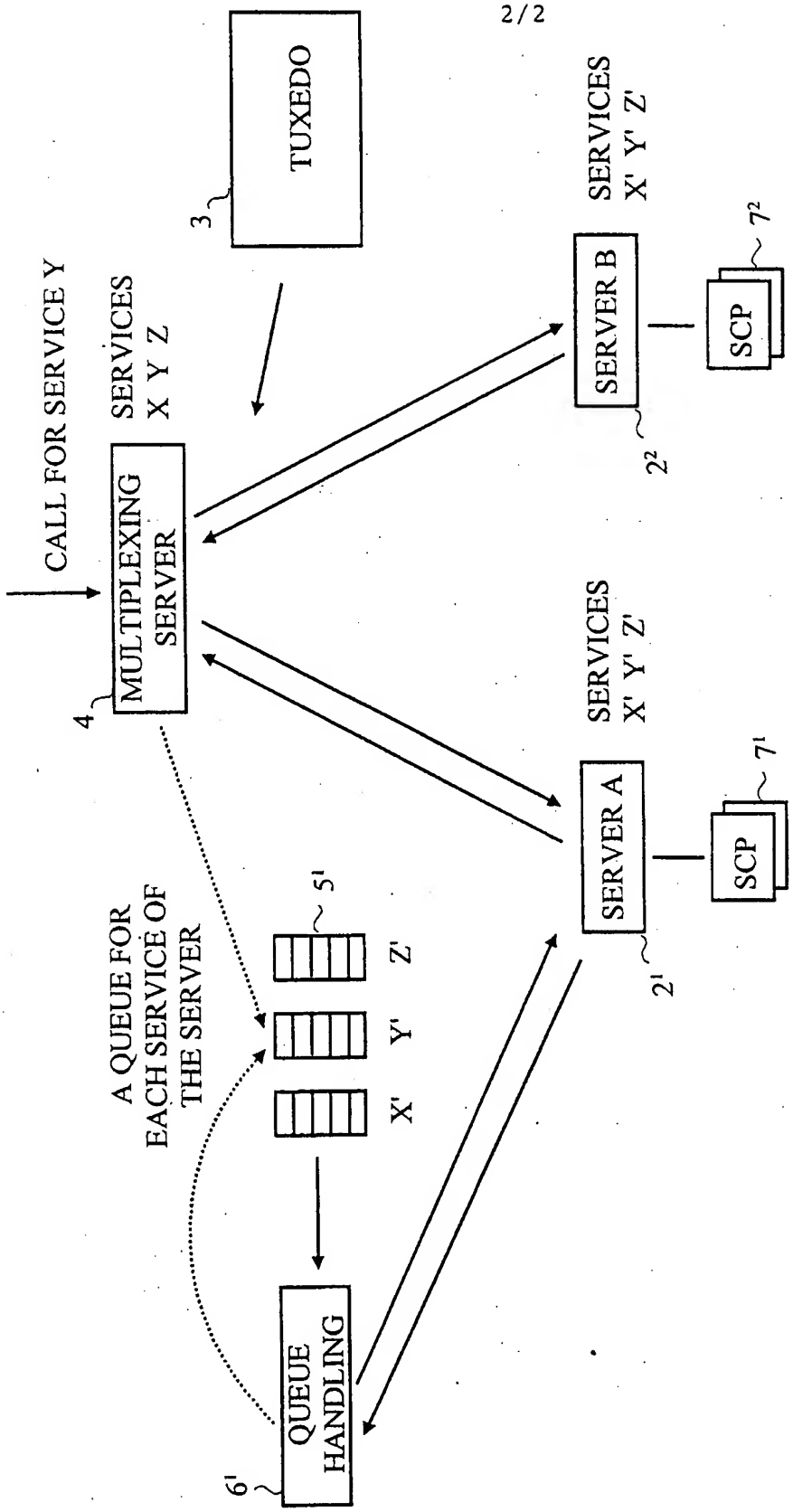

TUXEDO
3
CALL FOR SERVICE Y
SERVICES
X Y Z
MULTIPLEXING SERVER
4
SERVER B
2²
SCP
7²
SERVICES
X' Y' Z'
SERVICES
X' Y' Z'
SERVER A
2¹
SCP
7¹
A QUEUE FOR EACH SERVICE OF THE SERVER
5¹
X'
Y'
Z'
QUEUE HANDLING
6¹

Fig. 2